an application [having] derived from a program written in a high level programming language format, and

an interpreter operable to interpret such an application derived from a program written in a high level programming language format; and

a processor coupled to the memory, the processor configured to use the interpreter to interpret the application for execution and to use the communicator to communicate with the terminal.

6. (Amended) The integrated circuit card of claim 1, wherein

the application has been processed from a second application having a plurality of program elements, at least one being a string of characters, and

wherein in the first application the string of characters is [represented] replaced with [in the first application by] an identifier.

31. (Amended) A method for use with an integrated circuit card and a terminal, comprising:

storing an interpreter operable to interpret programs derived from programs written in a high level programming language and an application [having] derived from a program written in a high level programming language format in a memory of the integrated circuit card; and

using a processor of the integrated circuit card to use the interpreter to interpret the application for execution; and

using a communicator of the card when communicating between the processor and the terminal.

36. (Amended) The method of claim [1]31, wherein

the application has been processed from a second application having a plurality of program elements, at least one being a string of characters, further comprising:

[representing]replacing the string of characters in the first application [by]with an identifier.

58. (Amended) A smart card comprising:

a memory storing [a Java] an interpreter that interprets a subset of Java byte codes; and

a processor configured to use the interpreter to interpret [a Java] an application derived from a Java application program for execution.

Please cancel Claims 59 through 90 without prejudice.

91. (Amended) An integrated circuit for use with a terminal, comprising:

a communicator configured to communicate with the terminal;

a memory storing a first application that has been processed from a second application having a plurality of language elements including at least one [a] string of characters, the string of characters being [represented] replaced in the first application by an identifier; and

a processor coupled to the memory, the processor configured to use the interpreter to interpret the first application for execution and to use the communicator to communicate with the terminal.

93. (Amended) A method for use with an integrated circuit card and a terminal comprising:

processing a second application to create a first application, the second application having at least one programming element being a string of characters;

replacing [representing] the string of characters of the first application [by]with an identifier in the second application;

storing an interpreter and the first application in a memory of the integrated circuit card; and

using a processor of the integrated circuit card to use an interpreter to interpret the first application for execution.

95. (Amended) A microcontroller comprising:

a memory storing:

[an] a derivative application derived from an application having a class file format, and

an interpreter configured to interpret applications derived from applications having a class file format; and

a processor coupled to the memory, the processor configured to use the interpreter to interpret the derivative application for execution.

103. (Amended) The method of claim [100]101, wherein the creating includes using a Java interpreter.

104. (Amended) The method of claim [101]102, wherein

the storing of the first application is performed in association with manufacture of the integrated circuit card; and

the storing of the second application is performed at a later time after the manufacture is completed.

105. (Amended) An integrated circuit card for use with a terminal, comprising:

a communicator configured to communicate with the terminal;

a memory storing:

applications, each application derived from applications having a high level programming language format, and

an interpreter operable to interpret applications derived from applications having a high level programming language format; and

a processor coupled to the memory, the processor configured to:

a.) use the interpreter to interpret the applications for execution,

b.) use the interpreter to create a firewall to isolate the applications from each other, and

c.) use the communicator to communicate with the terminal.

[THIS IS FROM THE PRELIMINARY AMENDMENT] Please add the following new claims:

--106. A microcontroller having a set of resource constraints and comprising:

a memory, and

an interpreter loaded in memory and operable within the set of resource constraints,

the microcontroller having: at least one application loaded in the memory to be interpreted by the interpreter, wherein the at least one application is generated by a programming environment comprising:

a) a compiler for compiling application source programs in high level language source code form into a compiled form, and

b) a converter for post processing the compiled form into a minimized form suitable for interpretation by the interpreter.

107. The microcontroller of Claim 106, wherein the compiled form includes attributes, and the converter comprises a means for including attributes required by the interpreter while not including the attributes not required by the interpreter.

108. The microcontroller of Claim 106 wherein the compiled form is in a standard Java class file format and the converter accepts as input the compiled form in the standard Java class file format and produces output in a form suitable for interpretation by the interpreter.

109. The microcontroller of Claim 106 wherein the compiled form includes associating an identifying string for objects, classes, fields, or methods, and the converter comprises a means for mapping such strings to unique identifiers.

110. The microcontroller of Claim 109 wherein each unique identifier is an integer.

111. The microcontroller of Claim 109 wherein the mapping of strings to unique identifiers is stored in a string to identifier map file.

112. The microcontroller of Claim 106 where in the high level language supports a first set of features and a first set of data types and the interpreter supports a subset of the first set of features and a subset of the first set of data types, and wherein the converter verifies that the compiled form only contains features in the subset of the first set of features and only contains data types in the subset of the first set of data types.

113. The microcontroller of Claim 109 wherein the compiled form is in a byte code format and the converter comprises means for translating from the byte codes in the compiled form to byte codes in a format suitable for interpretation by the interpreter by:

using at least one step in a process including the steps:

a) recording all jumps and their destinations in the original byte codes;
b) converting specific byte codes into equivalent generic byte codes or vice-versa;
c) modifying byte code operands from references using identifying strings to references using unique identifiers; and
d) renumbering byte codes in the compiled form to equivalent byte codes in the format suitable for interpretation; and

relinking jumps for which destination address is effected by conversion step a), b), c), or d).

114. The microcontroller of Claim 106 wherein the application program is compiled into a compiled form for which resources required to execute or interpret the compiled form exceed those available on the microcontroller.

115. The microcontroller of Claim 106 wherein the compiled form is designed for portability on different computer platforms.

116. The microcontroller of Claim 106 wherein the interpreter is further configured to determine, during an interpretation of an application, whether the application meets a security criteria selected from a set of rules containing at least one rule selected from the set:

not allowing the application access to unauthorized portions of memory,

not allowing the application access to unauthorized microcontroller resources,

wherein the application is composed of byte codes and checking a plurality of byte codes at least once prior to execution to verify that execution of the byte codes does not violate a security constraint.

117. The microcontroller of Claim 106 wherein at least one application program is generated by a process including the steps of:

prior to loading the application verifying that the application does not violate any security constraints; and

loading the application in a secure manner.

118. The microcontroller of Claim 117 wherein the step of loading in a secure manner comprises the step of:

verifying that the loading identity has permission to load applications onto the microcontroller.

119. The microcontroller of Claim 117 wherein the step of loading in a secure manner comprises the step of:

encrypting the application to be loaded using a loading key.

120. A method of programming a microcontroller having a memory and a processor operating according to a set of resource constraints, the method comprising the steps of:

inputting an application program in a first programming language;

compiling the application program in the first programming language into a first intermediate code associated with the first programming language, wherein the first intermediate code being interpretable by at least one first intermediate code virtual machine;

converting the first intermediate code into a second intermediate code; wherein the second intermediate code is interpretable by at least one second intermediate code virtual machine; and

loading the second intermediate code into the memory of the microcontroller.

121. The method of programming a microcontroller of Claim 120 wherein the step of converting further comprises:

associating an identifying string for objects, classes, fields, or methods; and

mapping such strings to unique identifiers.

122. The method of Claim 121 wherein the step of mapping comprises the step of mapping strings to integers.

123. The method of Claim 120 wherein the step of converting comprises at least one of the steps of:

a) recording all jumps and their destinations in the original byte codes;
b) converting specific byte codes into equivalent generic byte codes or vice-versa;
c) modifying byte code operands from references using identifying strings to references using unique identifiers;
d) renumbering byte codes in a compiled format to equivalent byte codes in a format suitable for interpretation; and
e) relinking jumps for which destination address is effected by conversion step a), b), c), or d).

124. The method of Claim 120 wherein the step of loading the second intermediate code into the memory of the microcontroller further comprises checking the second intermediate code prior to loading the second intermediate code to verify that the

second intermediate code meets a predefined integrity check and that loading is performed according to a security protocol.

125. The method of Claim 124 wherein the security protocol requires that a particular identity must be validated to permit loading prior to the loading of the second intermediate code.

126. The method of Claim 124 further characterized by providing a decryption key and wherein the security protocol requires that the second intermediate code is encrypted using a loading key corresponding to the decryption key.

127. A microcontroller operable to execute derivative programs which are derivatives of programs written in an interpretable programming language having a memory and an interpreter, the microcontroller comprising:

(a) the microcontroller operating within a set of resource constraints including the memory being of insufficient size to permit interpretation of programs written in the interpretable programming language; and

(b) the memory containing an interpreter operable to interpret the derivative programs written in the derivative of the interpretable language wherein a derivative of a program written in the interpretable programming language is derived from the program written in the interpretable programming language by applying at least one rule selected from a set of rules including:

(1) mapping strings to identifiers;

(2) performing security checks prior to or during interpretation;

(3) performing structural checks prior to or during interpretation; and

(4) performing semantic checks prior to or during interpretation.

128. The microcontroller of Claim 127 wherein the derivative programs are class files or derivatives of class files.

129. The microcontroller of Claim 127 further comprising:
the memory containing less than 1 megabyte of storage.

130. The microcontroller of Claim 127 wherein the security checks the microcontroller is further comprising:

(c) logic to receive a request from a requester to access one of a plurality of derivative programs;

(d) after receipt of the request, determine whether the one of a plurality of derivative programs complies with a predetermined set of rules; and

(e) based on the determination, selectively grant access to the requester to the one of the plurality of applications.

131. The microcontroller of Claim 130, wherein the predetermined rules are enforced by the interpreter while the derivative program is being interpreted by determining whether the derivative program has access rights to a particular part of memory the derivative program is attempting to access.

132. The microcontroller of Claim 127 further wherein the microcontroller is configured to perform at least one security check selected from the set having the members:

(a) enforcing predetermined security rules while the derivative program is being interpreted, thereby preventing the derivative program from accessing unauthorized portions of memory or other unauthorized microcontroller resources,

(b) the interpreter being configured to check each bytecode at least once prior to execution to determine that the bytecode can be executed in accordance with pre-execution and post-execution checks, and

(c) the derivative program is checked prior to being loaded into the microcontroller to verify the integrity of the derivative program and loading is performed according to a security protocol.